22
150 pts.

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

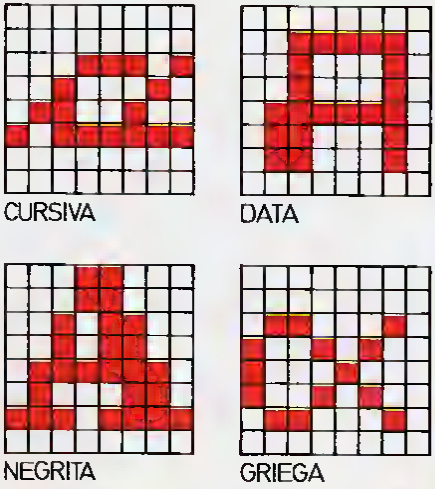
CARACTERES GRAFICOS

NO cabe duda de que el Spectrum cuenta con interesantes posibilidades en el campo de la representación de gráficos en la pantalla. Ya conocemos por comentarios anteriores, la existencia de los gráficos predefinidos por el Sistema, que son los contenidos en los números de la primera fila del teclado (1 a 8).

Pero como veremos más adelante, el ordenador nos da la facilidad de definir también nuestros propios caracteres. Esta es realmente una cualidad muy a tener en cuenta, puesto que nos permite la presentación altamente estética de todo tipo de programas, desde los técnicos, que precisan de caracteres especiales para su simbología (letras griegas, subindice y superíndices, etc...), hasta los de juego, en los cuales podremos definir nuestros propios personajes, naves espaciales, etc... además del acceso a la escritura en otros idiomas, pudiendo redefinir ciertos caracteres especiales, como las vocales acentuadas francesas, las diéresis germánicas, etc...

Por último, una técnica muy especial, conocida como REUBICACION DEL AREA GENERADORA DE CARACTERES, nos permitirá la redefinición de alfabetos completos (cirílico, griego, etc...) o bien la remodelación del habitual, obteniéndose distintos tipos de letra: cursiva, data o magnetic, princetown, etc... o cualquier otra a nuestro antojo.

En primer lugar, estudiaremos en profundidad la utilización de los gráficos predefinidos, aquellos que ya conocemos, para a continuación pasar a ver los sistemas de generación de nuestros propios caracteres.



La reubicación del generador de caracteres nos permitira la generacion de alfabetos completos.

Los caracteres definibles por el usuario (U.D.G.) nos dejan una puerta abierta para la erración de simbología especial (subindices, superíndices, acentos, diéresis, etc...).

GRAFICOS PREDEFINIDOS

Este juego de ocho caracteres gráficos, se ve en la realidad desdoblado en dieciséis, puesto que podemos obtener, con la sola pulsación de una tecla, los inversos de ellos *(INVERSE VIDEO)*. Para acceder a este juego de gráficos convencional, debemos recordar situarnos previamente en

¡!

Los graficos predefinidos se desdoblan en 16 caracteres: 8 en video normal y 8 en video inverso.

*

Para acceder al juego de caracteres gráficos, hemos de situar el cursor en el modo **GRAPHICS**, que se obtiene pulsando, o bien la tecla **GRAPH** (en el Plus), o bien las teclas **CAPS SHIFT + 9** (en el ZX Spectrum).

*

Los gráficos predefinidos son aquellos caracteres con codigos comprendidos entre el 128 y el 143.

el modo gráfico (cursor en modo G), mediante la pulsacion simultánea de **CAPS SHIFT + 9**, para después introducir, con o sin **CAPS SHIFT**, cualquiera de los caracteres numericos del 1 al 8 Dentro del modo G, los gráficos con **CAPS SHIFT** adoptan la configuración que aparece serigrafiada en blanco sobre las teclas; sin **CAPS SHIFT**, se accede al segundo juego, compuesto por los inversos del primero Esto que acabamos de afirmar es sólo válido para el modelo ZX Spectrum, si bien no para el Plus, en el cual las correspondencias de pulsación de teclas y carácter gráfico obtenido es exactamente la contraria

Si nos atenemos a esta regla, no tendremos nunca problemas con la selección de los gráficos No obstante, debemos recordar que normalmente se imprime en la pantalla en negro sobre blanco, y que la serigrafía del teclado es en blanco sobre gris en el modelo convencional, y en blanco sobre negro en el Plus Esto implica que las zonas de color blanco del carácter gráfico aparecerán en la pantalla de color negro y viceversa, para el ZX Spectrum, y que la correspondencia entre el color del teclado y el de la pantalla será la directa en el caso del Plus

La capacidad gráfica de los caracteres gráficos predefinidos, va más allá de lo que a simple vista podemos apreciar, tan solo un poco de experiencia en su manejo, nos puede reportar sorpresas muy agradables

Su utilización más común se da en la presentación de mensajes de grandes dimensiones en la pantalla Por otra parte, estos caracteres se pueden emplear también para la representación de funciones en baja resolución, es decir, aquellas en las cuales cada punto representado coincide con un cuarto de carácter

Como es lógico, los gráficos predefinidos forman parte integrante del juego de caracteres, debido a lo cual tienen asignado un código determinado Gracias a ello, estos caracteres pueden ser obtenidos no sólo por el teclado, sino también a traves de un programa, mediante la función **CHR$** Los códigos de los caracteres gráficos predefinidos van desde 128 al 143 El siguiente miniprograma nos los muestra en la pantalla, acompañados de su código correspondiente

```
10 PRINT "CODIGO","CARACTER"
20 PRINT "------","--------"
30 FOR I=128 TO 143
40 PRINT "  ";I,"     ";CHR$ I
50 NEXT I
```

En el programa podemos observar claramente la ventaja que supone poder emplear la función **CHR$**, puesto que ello nos permite utilizar una sola línea de programa para la impresión de caracteres (la 40), utilizando una variable (I) para designar el carácter concreto a representar

El modo de cursor G y las teclas numéricas del 1 al 8, dan acceso a los gráficos predefinidos

PECULIARIDADES DE LOS CARACTERES GRAFICOS

Los caracteres gráficos no pertenecen al código A S C I I estándar, y por tanto, la mayoría de las impresoras, a excepción de las especialmente di-

El juego de ocho caracteres predefinidos se desdobla en dieciseis

TECLA / CAMBIO	1	2	3	4	5	6	7	8
CON CAPS SHIFT	▙	▟	▄	▙	▌	▚	▗	■
SIN CAPS SHIFT	▛	▜	▄	▜	▐	▞	▟	□

señadas para nuestro ordenador, carece de la posibilidad de impresión de estos caracteres. Debido a ello, a la hora de realizar los listados en papel de los programas, se utiliza un convenio que nos interesa conocer, ya sea para poder interpretar los listados hecho por otros, fundamentalmente los aparecidos en publicaciones, como para utilizarlos en nuestros propios programas que deseemos enviar a concursos, o a nuestros amigos.

Dado que en un listado nunca puede aparecer, por la vía normal, un carácter subrayado, este tipo de distinción es la que se emplea para designar los gráficos. Así pues, siempre que veamos en un listado un carácter subrayado, debemos interpretar que se trata del gráfico de la tecla cuyo carácter principal aparece subrayado.

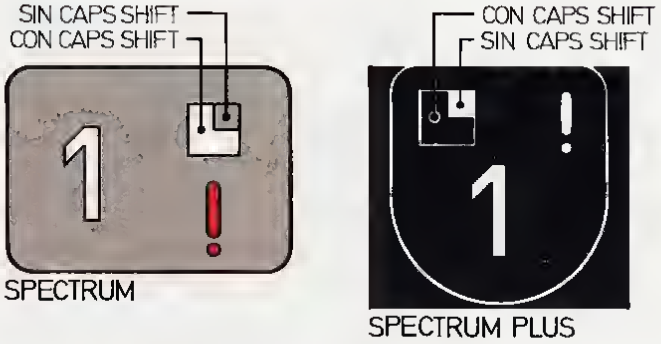
Así por ejemplo, un número cinco (5) subrayado, equivale al gráfico que se obtiene por la pulsación de la tecla **5**. Por tanto, para su introducción deberemos proceder de la siguiente forma:

— Pasar el cursor a modo gráfico (cursor **G**). En el Spectrum Plus existe una tecla especial a tal fin (**GRAPH**), mientras que en el caso del ZX Spectrum es necesaria la pulsación de dos teclas simultáneamente: **CAPS SHIFT** + **9**.

— A continuación, pulsaremos la tecla cuyo carácter principal aparece subrayado. En el caso concreto de nuestro ejemplo, deberemos pulsar la tecla **5**, lo cual hará aparecer en la pantalla no dicho dígito, sino su carácter gráfico correspondiente.

— Por último, y en caso de que no haya más caracteres gráficos consecutivos que introducir, deberemos restablecer el modo normal del cursor, ya sea L o C, para poder continuar con la introducción del listado. Para ello, deberemos pulsar de nuevo las teclas **CAPS SHIFT** + **9**, o bien únicamente la del **9**. En el caso del modelo Plus, a esta posibilidades se añade la de pulsación de la tecla **GRAPH**.

Hasta ahora el sistema de subrayado ha cumplido plenamente su cometido, pero los problemas llegan de nuevo cuando deseamos representar los caracteres gráficos predefinidos en vídeo inverso, es decir, aquellos que se obtienen en el modo gráfico mediante la pulsación simultánea de una tecla numérica (de 1 a 8) y la tecla **CAPS SHIFT**.

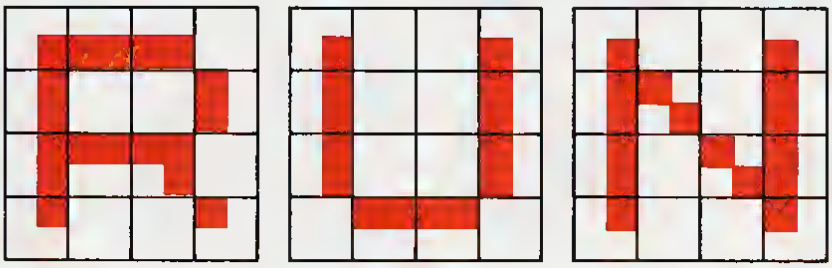


La correspondencia entre la serigrafía del teclado y el gráfico predefinido que se obtiene mediante la pulsación de las teclas correspondientes, es inversa en los modelos Plus y ZX Spectrum.

Este tipo de gráficos que habitualmente se conocen como GRAFICOS CAMBIADOS (*SHIFT* en inglés se traduce por CAMBIO), se representan de forma similar a los anteriores, pero esta vez con un subrayado doble. Así por ejemplo, un carácter 8 doblemente subrayado en un listado, debe ser introducido pasando el cursor a modo **GRAPHICS** (cursor G) y pulsando simultáneamente las teclas **CAPS SHIFT** + **8**. Nunca olvidemos desconectar el modo gráfico al finalizar la introducción de una serie de los mismos, es decir, cuando el próximo carácter a introducir ya no deba ser considerado como gráfico.

A continuación adquiriremos los conocimientos sobre la forma en que el Spectrum almacena el diseño de los caracteres en su memoria. La zona de memoria destinada a tal fin, se denomina AREA GENERADORA DE CARACTERES. Pues bien, es interesante destacar que los caracteres gráficos predefinidos no se encuentran en dicha

Los gráficos predefinidos se suelen utilizar para la presentación de mensajes en grandes dimensiones



¡!

Los caracteres gráficos no están contemplados por el código ASCII estandar, y por tanto la mayoría de las impresoras no los reconocen. Para soslayar este inconveniente, se suele adoptar el convenio de subrayar el carácter principal de la tecla en la cual se encuentre el grafico.

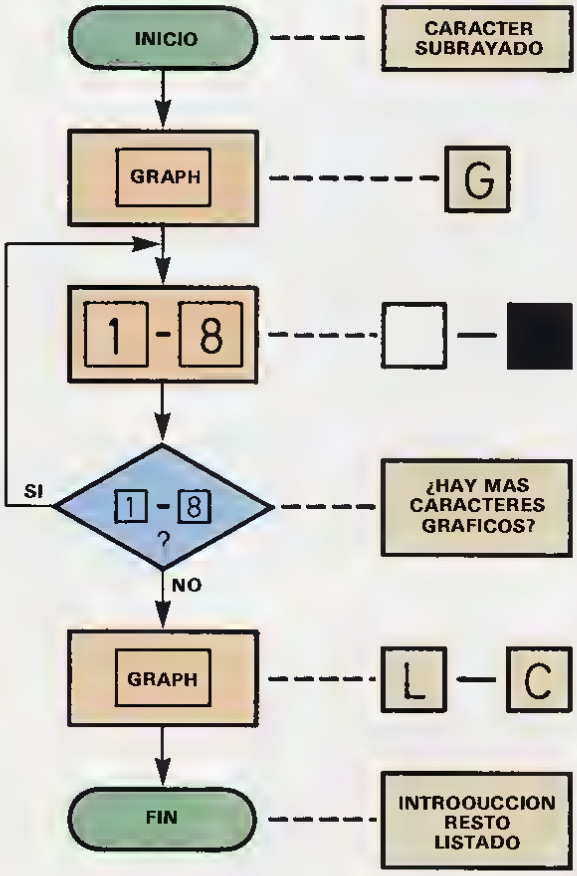
*

La representación en pantalla de cualquier carácter de control es un signo de interrogación.

zona, ni realmente en ninguna otra, sino que cada vez que es necesaria su impresión, el ordenador calcula su configuración en base al código que reciben

Por este motivo, algunas subrutinas de caracteres gigantes, como la que hemos utilizado en ocasiones en la sección de PROGRAMA (Psion Computers, cinta de demostración HORIZONTES), no pueden representar este tipo de caracteres, puesto que se basan en la ampliación de la forma que adoptan los caracteres en el área generadora, y los gráficos predefinidos no se hayan reflejados en ella.

Secuencia de introducción de un carácter subrayado



Cuando en un listado encontremos un caracter subrayado, habremos de introducir el gráfico de la tecla cuyo carácter principal se subraya

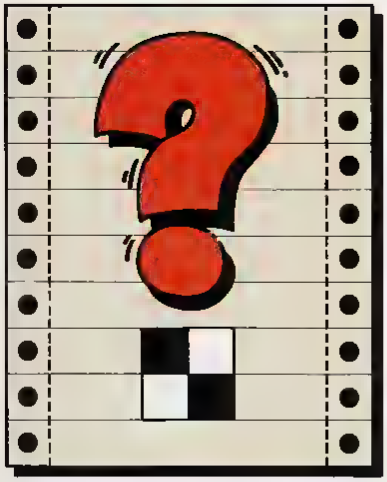
EL GENERADOR DE CARACTERES

Al hablar de la composición de la pantalla, dijimos que es similar a la de un papel milimetrado, en el cual podemos contar 24 filas de 32 cuadrículas. Digamos también, para los iniciados en el álgebra, que es como una matriz de 24 filas por 32 columnas.

En cualquier caso, cada una de las 768 (24×32) posiciones de carácter en las cuales se divide la pantalla, consta a su vez de una configuración de puntos, concretamente se trata de una cuadrícula de 8 por 8, es decir, 64 puntos en total. Esta cuadrícula elemental de 8 por 8, permite la representación del juego completo de caracteres del Spectrum, alfabéticos, numéricos y gráficos, donde el atributo **INK** afecta a los puntos encendidos del carácter, y el atributo **PAPER** a los apagados.

En el área generadora de caracteres, se encuentran memorizadas las formas de cada uno de los caracteres representables por el ordenador desde el de código 32 (espacio), hasta el símbolo del *copyright* (©), correspondiente al código 127. Como bien sabemos, por debajo de 32 se encuentran los caracteres de control no representables, y por encima de 127 los gráficos y los TOKEN (palabras BASIC multicarácter).

Pues bien, para la definición de nuestros propios caracteres habremos de seguir un sistema de memorización idéntico al del generador de caracte-



La mayoría de las impresoras, no diseñadas especificamente para nuestro aparato, ignoran los caracteres gráficos.

res, debido a lo cual hemos de estudiar previamente éste.

En primer lugar, diremos que el generador se encuentra en los últimos 3/4 K de la memoria ROM (768 bytes), la reservada al Sistema, en la cual nunca podemos escribir y que nunca se borra, ni tan siquiera apagando el ordenador. Esto es algo muy sencillo de intuir, dado que si no fuera así, una vez apagado el Spectrum éste no sabría, al encenderlo nuevamente, la forma de las letras, y no podría escribir caracteres en la pantalla para comunicarse con nosotros.

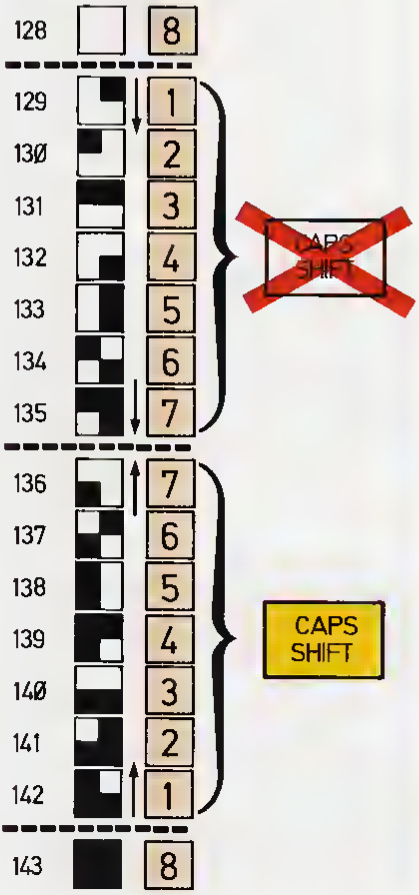
Así pues, el generador de caracteres se encuentra entre las direcciones de memoria 15616 (3D00 en base hexadecimal) y 16383 (3FFF hexadecimal), ambas inclusive. Este dato será muy importante cuando estudiemos el sistema de reubicación del generador de caracteres.

La forma en la cual se encuentran almacenados los caracteres es bastante fácil de asimilar. Cada carácter está compuesto por ocho líneas de ocho puntos, cada uno de los cuales puede adoptar dos únicos estados: encendido (color de tinta) o apagado (color de fondo). Por tanto, cada punto puede ser representado mediante una unidad mínima de información (bit), para lo cual se ha adoptado el convenio de designar con un cero los puntos apagados, y con un uno los encendidos. Así pues, si cada punto es un bit, y una línea tiene ocho puntos, una línea de cualquier carácter se puede representar mediante ocho bits, es decir, con un byte. Dado que los caracteres completos están constituidos por ocho líneas, en la memoria podemos representarlos por ocho bytes consecutivos.

Así por ejemplo, el carácter ESPACIO está memorizado con los ocho bytes (64 bits) que van desde 15616 hasta 15623 (ambos inclusive, la ADMIRACION, está almacenada desde 15624 hasta 15631, etc... Cada ocho bytes consecutivos, a partir del comienzo del generador de caracteres, representan la forma de un carácter completo, visualizados en la pantalla como sus ocho líneas, una debajo de otra.

Sabiendo unas poquitas matemáticas (de eso el ordenador entiende bastante), conociendo el punto de inicio del generador de caracteres y el código del carácter a buscar, al ordenador no le es nada difícil localizar la dirección del generador de caracteres, a partir de la cual se haya la for-

A los gráficos predefinidos les corresponden los códigos comprendidos entre 128 y 143



¡!

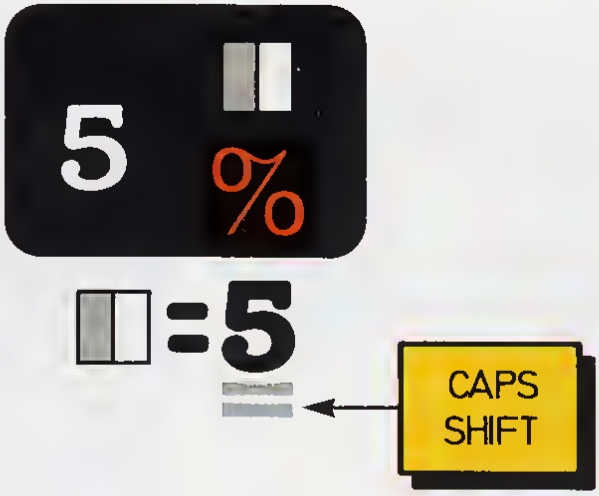
Los graficos predefinidos no se encuentran en el área generadora de caracteres en ROM, siempre que son necesarios, una rutina del *firmware* se encarga de construirlos en base a su código

*

Los caracteres que aparecen doblemente subrayados corresponden a los graficos cambiados de las teclas cuyo caracter principal es el afectado por el subrayado. Se denominan graficos cambiados a aquellos obtenidos pulsando la tecla correspondiente en combinación con **CAPS SHIFT**

*

El area generadora de caracteres en ROM contiene las configuraciones de los caracteres entre los codigos 32 (espacio) y 127 (copyright)



Los caracteres subrayados doblemente corresponden a los gráficos cambiados de las teclas

Como hemos visto hace tan solo unos instantes, las formas de los caracteres se almacenan en una zona de la memoria denominada generador de caracteres, sin embargo, no todos ellos se encuentran reflejados en ella, sino únicamente lo de códigos comprendidos entre 32 y 127. ¿Qué sucede con los restantes?

Los primeros treinta y dos (del 0 al 31), no suponen ningún problema, puesto que se trata de caracteres de control, sin forma determinada, y que en caso de verse en la obligación de imprimirlos (por ejemplo, **PRINT CHR$ 0**), los sustituye por un signo de interrogación.

A partir del 165 se encuentran las TOKEN, que lógicamente no tienen representación en el generador de caracteres, puesto que se componen de varios caracteres simples. Del mismo modo, los gráficos predefinidos, que abarcan desde el código 128 hasta el 143 (ambos inclusive), son generados por el Spectrum cuando es necesaria su utilización, debido a lo cual tampoco están presentes en el área de caracteres. Ahora bien, para la definición de los gráficos del usuario, es de máximo interés que conozcamos su situación en la memoria.

Como es lógico, no se encuentran dentro del generador de caracteres, puesto que ello simplicaría que se hallaran en la ROM, y por tanto, que

¡!

ma del carácter a representar. **Dirección=15616+(Código-32)*8**, para todo código entre 32 y 127.

Se denominan TOKEN a las palabras BASIC multicarácter, las cuales ocupan los últimos 128 codigos del juego ASCII del Spectrum.

*

Los bits a uno en un caracter, corresponden a los puntos encendidos de la pantalla, es decir, los que se presentan en el color de primer termino (INK).

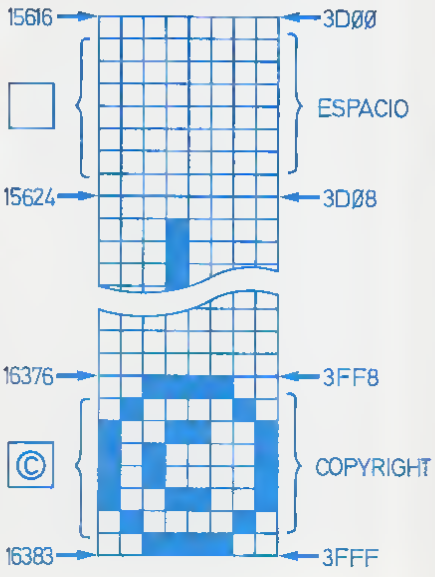
CARACTERES GRAFICOS DEFINIBLES

Además de los dieciséis caracteres gráficos predefinidos, el Spectrum tiene veintiuno más disponibles, para definirlos con la configuración exacta que decidamos. Para acceder a ellos, basta con situar el cursor en el modo gráfico (G), y pulsar una tecla de la A a la U.

Si antes de efectuar ninguna definición de caracteres, pretendemos acceder a un gráfico definible, nos encontraremos con que su configuración es exactamente la misma que la correspondiente al carácter alfabético que representa.

Esto es debido a que el Spectrum supone, como valores por defecto para cada una de las teclas de los gráficos definibles, el carácter principal que representa su tecla. Ahora veremos por qué

El generador de caracteres en ROM comienza en la dirección 15616 y acaba en 16383

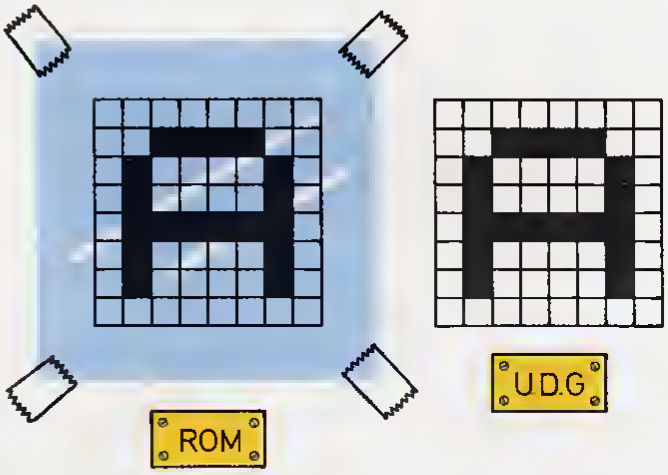


su configuración fuera inalterable, lo cual va contra su propia esencia. Así pues, estos se deben localizar en algún lugar de la memoria RAM (disponible para el usuario).

Los 168 bytes que ocupan (21 gráficos por ocho bytes cada uno), se encuentran situados concretamente al final de la memoria. En el modelo de 16 K a partir de la dirección 32600 (hasta 32767) y en el de 48 K o Plus a partir de 65368 (hasta 65535). Como ya hemos anticipado, la forma en la cual se almacenan estos caracteres dentro de su zona es la misma que la de los caracteres predefinidos en el generador en ROM: 8 bytes consecutivos por carácter.

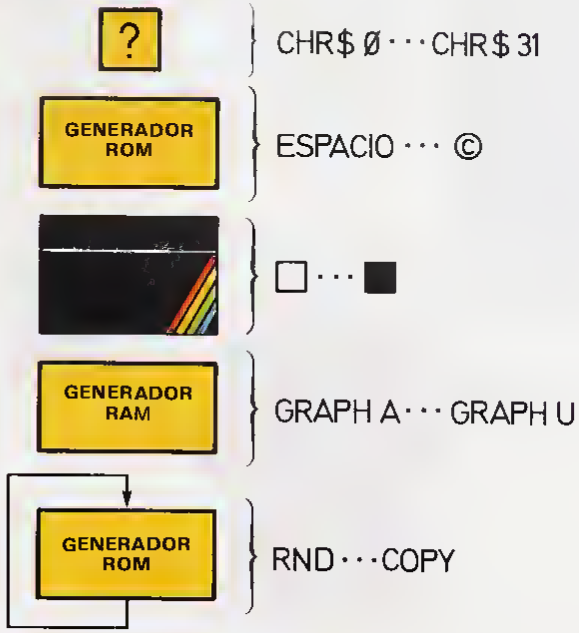
Como es lógico, al estar estos caracteres ubicados en la RAM, cuando se enciende el ordenador no tienen ningún valor inicial (la RAM se borra al desconectar el Spectrum de la corriente). Por ello, entre las tareas que el ordenador lleva a cabo nada más conectarse, antes de hacer aparecer el mensaje de presentación, se cuenta la de dar una forma inicial a los gráficos definibles, copiando el generador de caracteres desde la A hasta la U, aunque podremos alterar estos valores iniciales sin ningún problema.

Así pues, para la definición de los caracteres nos apoyaremos fundamentalmente en dos puntos, la obtención de los bytes que configuran el carácter tal como lo diseñamos, y la modificación del área de memoria que los alberga. A tal fin, el BASIC dispone de dos sentencias especiales que nos facilitan este trabajo: **BIN** y **POKE**.



Aunque en un principio los UDG tengan la misma configuración que los caracteres de la ROM, pueden ser alterados con toda libertad

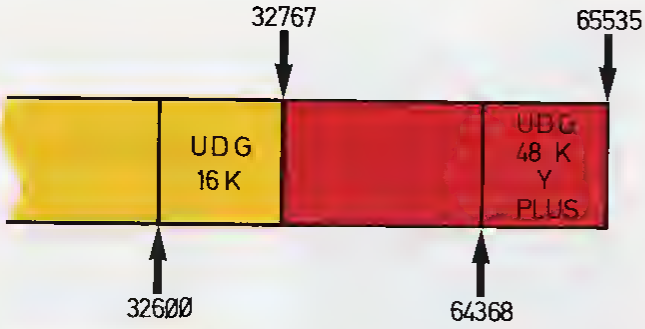
Los caracteres del A.S.C.I.I. del Spectrum se obtienen de muy diversas maneras



BIN

El empleo de la función **BIN** permite introducir directamente en el ordenador números en formato binario, en vez de en la forma habitual (decimal). Como la definición del caracter abarca 8 líneas, será necesario teclear la configuración binaria de todas ellas, para definirlo completamente.

Cada una de las posiciones de memoria (líneas del carácter), puede adoptar valores entre 00000000 y 11111111 binario, cuya correspondencia en decimal es 0 y 255, respectivamente. Está claro que la notación es más conveniente para la definición de caracteres, puesto que se adapta mucho mejor a la forma en que se componen éstos. Sin embargo, conviene a veces, superada la fase de definición, incluir los datos de formación de los caracteres en los programas en el formato decimal, dado que éste resulta más corto y fácil de recordar.



Las áreas de UDG (gráficos definibles por el usuario) se encuentran en diferentes lugares en el modelo de 16 y 48 K o Plus

¡!

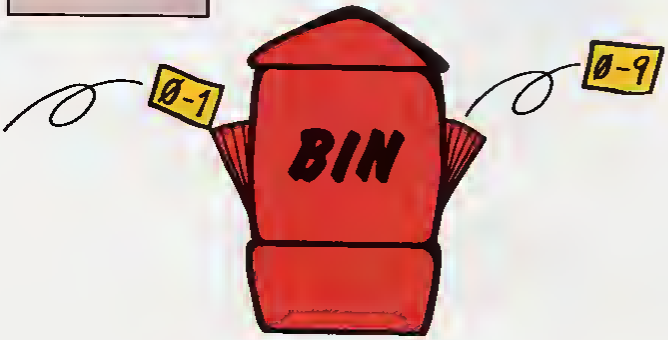
Los bits a cero en un carácter, corresponden a los puntos apagados de la pantalla es decir los que se presentan en el color de fondo (**PAPER**)

*

Un carácter viene definido en la memoria por una serie de ocho bytes consecutivos, cada uno de los cuales representa una línea del mismo Dentro de cada byte (línea) los bits indican el estado de un punto de la pantalla

El formato de la función **BIN** es sencillo. Consta de la propia palabra y un argumento de 1 a 16 dígitos binarios (deben ser forzosamente unos o ceros), lógicamente, como toda función BASIC, **BIN** habrá de ir precedida de una palabra clave. **PRINT BIN** realiza la conversión del argumento suministrado en binario, a una respuesta impresa en la pantalla en decimal Cuando no se expresa ningún argumento, el resultado de la conversión es 0 Por otra parte, cuando como argumento suministramos un número negativo, compuesto por cualquier serie de dígitos decimales, la operación **BIN** no se ejecuta, produciéndose un **PRINT** normal, en decimal, del número tecleado En cuanto al rango de operación, diremos que se encuentra entre 0 y 65535 decimal, o lo que es lo mismo, entre 0 y 16 bits, que corresponden a

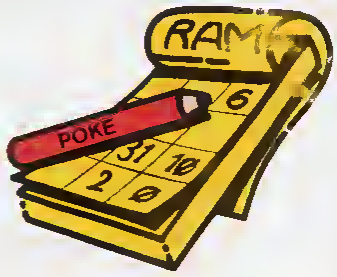
BIN es la función BASIC que permite traducir de binario a decimal



la ocupación de dos posiciones de memoria consecutivas, aunque para el caso de la definición de caracteres, sólo nos será necesario hacer uso de números binarios de 1 a 8 dígitos

POKE

Esta sentencia deposita un valor numérico entero entre 0 y 255, en la posición de memoria que deseemos (de 0 a 65535) Por tanto, modifica el contenido de la memoria en cualquier punto de la misma, depositando un nuevo byte
La dirección de memoria sobre la cual ejerce su



POKE *deposita un valor de la memoria*

acción **POKE** puede pertenecer a la ROM, puesto que no se producirá ningún error de ejecución, pero carecerá de efecto alguno Si se facilita una dirección fuera de límites, es decir, menor que cero o mayor que 65535, obtendremos el error B **Integer out of range** (entero fuera de rango) Sin embargo, la utilización de direcciones con decimales sí que está permitida, aunque el ordenador se encarga antes de depositar el byte, de redondear al entero más próximo la dirección facilitada
En cuanto al rango en el cual se mueve el byte a depositar, se encuentra entre –255 y 255, obteniéndose el error B **Integer out of range** al emplear valores fuera de este límite Como en el caso de las direcciones, los valores decimales se redondean al entero más próximo desde 0 5 en adelante, al superior, y hasta 0 49 , al inferior Por último, los valores negativos equivalen al complemento hasta 255, es decir, –1 equivale a 255, –2 a 254, etc hasta –255 que equivale a 1

ALGEBRA DE BOOLE

Ya en el capítulo anterior comenzamos nuestra introducción al tema del álgebra de Boole, observando el funcionamiento de la primera de las operaciones lógicas: AND.

A continuación pasaremos al estudio pormenorizado del resto de las funciones lógicas, lo cual nos brindará la base suficiente para comprender la relación de este área de las matemáticas con la informática, y más concretamente con la circuitería (*hardware*) de nuestro Spectrum. Pasemos por tanto a la siguiente operación lógica.

OR

Esta operación se representa mediante un signo más (+), pero debemos de tener mucho cuidado para no confundirla con la operación aritmética suma. Su tabla de verdad es la siguiente:

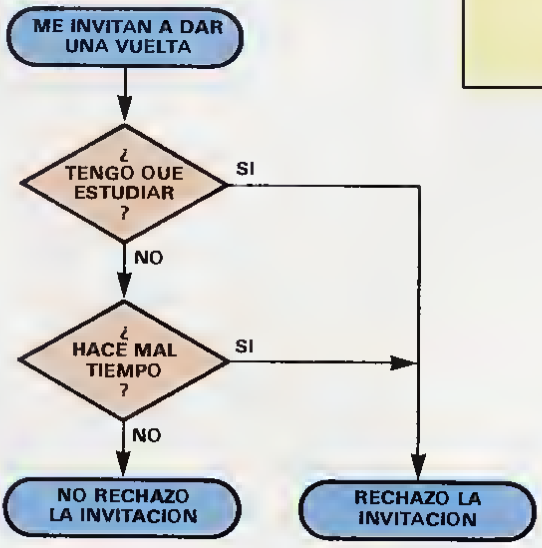
A + B	O
0 + 0	0
1 + 0	1
0 + 1	1
1 + 1	1

Dada la similitud de los resultados obtenidos por esta función con los de una suma aritmética, en el caso de que los valores fueran considerados como cuantificables (números) y no como valores lógicos, esta operación se conoce también bajo la denominación de SUMA LOGICA.

El nombre de OR, palabra inglesa cuyo significado es O, se asocia con el tipo de problemas lógicos sobre los cuales se aplica: SI <condición> O <condición> ENTONCES ... <resultado>. Así pues, si con AND es indispensable que sean ciertas ambas condiciones para que sea cierto el resultado, con OR es suficiente que cualquiera de las dos se cumpla, aunque no plantea ningún problema que las dos condiciones se cumplan. Veamos un ejemplo de este tipo de estructura lógica. Supongamos que un amigo nos propone ir a dar un paseo, el que rechacemos o no la invitación (resultado lógico) depende directamente de dos circunstancias (variables lógicas) que tengamos que estudiar y que haga buen tiempo.

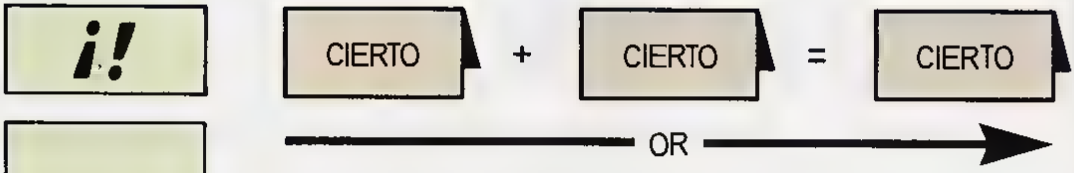
Llamaremos A a la variable lógica «tenemos que estudiar», que como es habitual puede adoptar dos valores: 1 (tenemos que estudiar) y 0 (no tenemos que estudiar). Por otra parte, la segunda variable lógica, B, corresponderá a la condición «hace buen tiempo», que adoptará los valores 1 o 0 según se cumpla o no, respectivamente. Al resultado de nuestro problema le denominaremos O, equivaliendo a «rechazamos la invitación», como es normal, el resultado podrá también adoptar dos únicos valores, 1 si rechazamos la invitación y 0 si no lo hacemos. Dado que es suficiente que cualquiera de las dos variables se cumpla para verificar el resultado, nos encontra-

En la estructura lógica del tipo OR, es suficiente que una de las condiciones se cumpla para que se verifique el resultado



¡!

La operación lógica AND se representa por un círculo relleno (●), OR por un signo más (+), XOR por un signo más inscrito en un círculo (⊕) y NOT por una raya horizontal sobre el operando (¯)



Para que se verifique un resultado de la operacion XOR sus operandos deben encontrarse en estados diferentes

*

NOT es una operación logica con un solo operando cuya misión es cambiar el estado de ésle
V

mos ante una operación OR. Sólo nos resta asignar los valores para cada caso, y hacer las correspondientes consultas en la tabla de verdad.
Si no tenemos que estudiar (A=0) y no hace mal tiempo (B=0) no rechazaremos la invitación (Q=0), puesto que 0 + 0 = 0. Si tenemos que estudiar (A=1) y no hace mal tiempo (B=0) rechazaremos la invitación (Q=1), dado que tenemos que quedarnos en casa para estudiar. 1 + 0 = 1.
Algo muy parecido ocurre en el siguiente caso: si no tenemos que estudiar (A=0) pero hace mal tiempo (B=1) rechazaremos la invitación (Q=1), puesto que 0 + 1 = 1. Por último, si ambas condiciones se cumplen, es decir, si tenemos que estudiar (A=1) y además hace mal tiempo (B=1), evidentemente rechazaremos la invitación (Q=1), efectivamente, 1 + 1 = 1. No olvidemos que el signo más (+) indica en estos casos una suma lógica y no una aritmética.
Sin duda las dos operaciones lógicas más comunes son **AND** y **OR**, sobre todo para los usuarios del BASIC, puesto que disponemos de dos operadores lógicos, con los mismos nombres y misiones que los aquí estudiados, los cuales nos permiten obtener este tipo de estructuras lógicas en combinación con la sentencia de decisión **IF**.

La operación OR es también conocida como SUMA LOGICA.

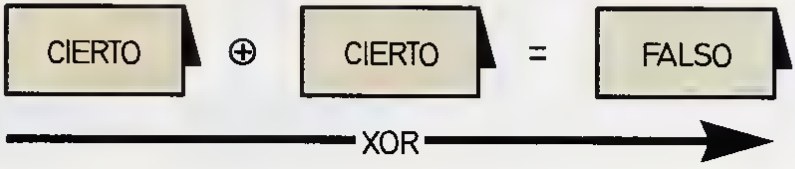
A⊕B	Q
0⊕0	0
1⊕0	1
0⊕1	1
1⊕1	0

Constituye por tanto una estructura en la cual as preciso que sólo una de las condiciones se cumpla para que el resultado se verifique (sea 1).
Con un ejemplo comprenderemos mejor su significado. Tenemos dos buenos amigos: Juan y Pedro; con cada uno de ellos nos llevamos a las mil maravillas, pero sin embargo, ellos no se aprecian, y siempre que se ven comienzan a discutir.
Nos avisan unos amigos de que van a salir a dar un paseo y nos proponen acompañarles.
Puesto que hemos terminado los exámenes y hace buen tiempo, el que aceptemos su invitación (resultado logico) depende de dos factores: la asistencia de Pedro (variable lógica A) y la asistencia de Juan (variable logica B). Pero de una manera un tanto especial.
Si no van ni Pedro ni Juan (A=0) y B=0), no aceptaremos la invitación (Q=0), puesto que nos aburriremos (0 + 0 = 0). Si va Pedro (A=1) y no va Juan (B=0), no rechazaremos la invitación (Q=1). Pedro es un buen amigo y nos divertiremos (1⊕0 = 0). Análogamente, si no asiste Pedro (A=0) pero sí Juan (B=1), aceptaremos la invitación.

XOR

Esta operación adopta su nombre del inglés eXclusive OR, cuya traducción literal es Q EXCLUSIVQ, y se representa mediante un signo más inscrito en un círculo (⊕). Su tabla de verdad es la siguiente:

En la tabla de verdad de la operación XOR, sólo dan como resultado uno los bits en diferente estado



(Q=1), dado que tampoco nos aburriremos (0 ⊕ 1 = 1) Por último, si van tanto Pedro (A=1) como Juan (B=1), la mezcla se hace explosiva y el disgusto seguro, debido a lo cual decidimos no ir con ellos (Q=0, puesto que 1 ⊕ 1 = 0) Así pues, hemos resuelto un problema lógico del tipo XOR

NOT

Esta es la última de las funciones lógicas, y se diferencia de las demás en que actúa sobre un solo operando Recibe su nombre (NOT en inglés significa NO) de la misión que le es encomendada, consistente en «negar» a su operando, es decir, cambiar su valor lógico, cualquiera que sea éste Su representación es una raya horizontal sobre el valor negado. La tabla de verdad de esta operación, al tener un solo operando, es la más simple de todas

$\overline{A}$	Q
$\overline{0}$	1
1	0

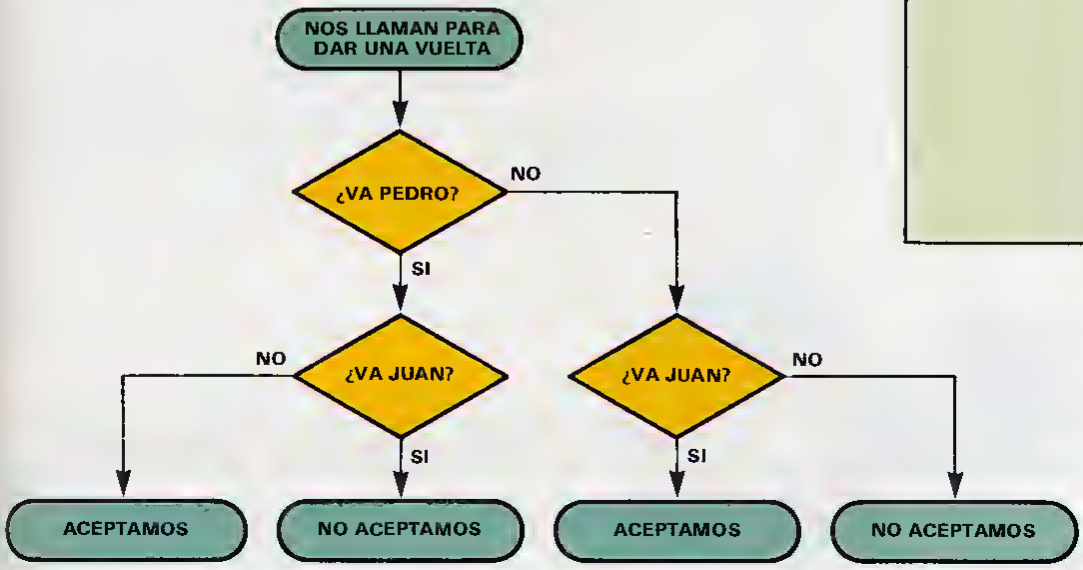
Esta función es la representación del más puro espíritu de la contradicción, y recuerda a los dos compañeros que no se llevan nada bien y se encuentran por la calle. «¿Hacia donde vas Andrés, hacia la derecha o hacia la izquierda?» Si Andrés va hacia la derecha, Manolo piensa ir hacia la izquierda, y si Andrés dice ir hacia la izquierda, Manolo se dirigirá a la derecha

Con NOT finalizamos la revisión de las operaciones lógicas, y a continuación pasamos a analizar la aplicación de las mismas a la informática

APLICACION DEL ALGEBRA DE BOOLE

Desde el punto de vista de la programación, la utilización habitual no es sobre un solo bit, tal como hemos visto en las tablas de verdad, sino sobre un byte Este problema de las operaciones

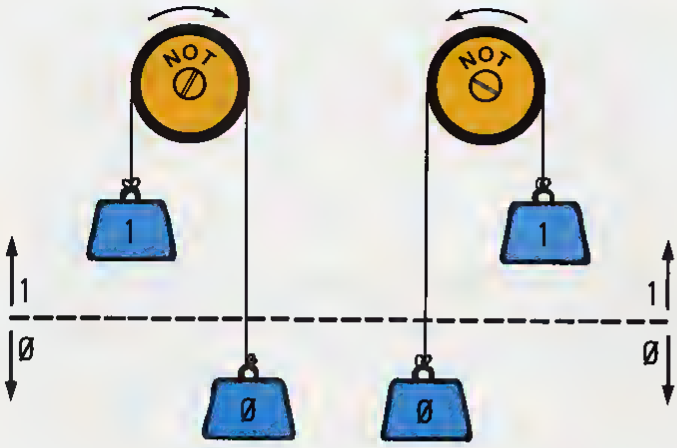
Para que se verifique el resultado en las estructuras XOR, es necesario que sólo una de las condiciones se cumpla



La operacion lógica OR se utiliza en las estructuras del tipo Si <condición> O <condicion> ENTONCES <resultados>

*

En la tabla de verdad para OR, todos los resultados dan uno, salvo cuando ambas condiciones son cero



La operación lógica NOT siempre cambia el estado de su operando

multibit, se resuelve realizando las operaciones simples de los bits que ocupen iguales posiciones en los dos bytes a operar.

Así por ejemplo, el resultado de 01010100 • 10010001 es 00010000, puesto que, 0•1=0, 1•0=0, 0•0=0, 1•1=1, 0•0=0, 1•0=0, 0•0=0 y 0•1=0.

La aplicación de estas funciones a la programación, fundamentalmente en código máquina, es muy importante. La principal tarea de AND es anular determinados bits de un byte, dejando los otros intactos. Así por ejemplo, realizar un AND de cualquier operando con el byte 11110000, asegura que sea cual sea éste, el resultado corresponderá a sus cuatro bits de la izquierda inalterados y los restantes con valor cero.

OR por el contrario, se emplea para asegurarse que determinados bits del resultado sean uno, y que el resto permanezcan inalterados. Por ejemplo, realizar un OR de cualquier operando con el byte 11110000, asegura que los cuatro bits de la izquierda del resultado serán unos, y que los cuatro menos significativos (los de la derecha) no se veran afectados.

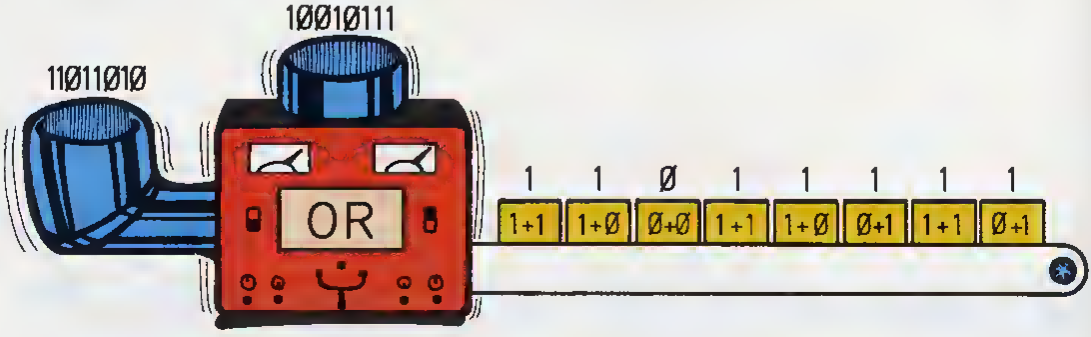
A estas técnicas basadas en AND y OR, consistentes en preparar uno de los operandos para que sea cual sea el otro, le anule o active determinados bits en el resultado dejando el resto intactos, se les denomina ENMASCARAMIENTOS.

XOR tiene la propiedad de que al efectuarlo dos veces consecutivas se retorna al operando inicial, es decir, A⊕B = O y O⊕B = A. Otra de sus propiedades es asegurar que el resultado será cero sean cuales sen A y B, siempre que se cumplan que son iguales: para todo A=B se cumple que A ⊕B = 0.

Por último, la misión de NOT es claramente obtener un resultado opuesto al del operando, cualquiera que sea éste. Esta función, hablando en términos de programación en código máquina, se suele conocer como complementación, y es de vital importancia en el sistema que se emplea para las operaciones aritméticas con bytes, en las cuales se tenga en cuenta el signo, la denominada aritmética en complemento a dos, que por el momento no es necesario que conozcamos.

Pero sin duda alguna, la aplicación más interesante del álgebra de Boole la tenemos en la propia circuiteria del Spectrum, dado que en ella se basa el núcleo «inteligente» de los ordenadores. Antes de pasar al siguiente capítulo, donde realizaremos un estudio sobre el funcionamiento general de este núcleo «inteligente» debemos conocer una última regla de oro del álgebra de Boole.

Las operaciones logicas byte a byte, se efectuan como operaciones bit a bit múltiples entre los bits que ocupan las mismas posiciones dentro de los operandos.



AND

1 0 1 0 1 0 1 0 • 1 1 1 1 0 0 0 0 = 1 0 1 0 0 0 0 0

OR

1 0 1 0 1 0 1 0 + 1 1 1 1 0 0 0 0 = 1 1 1 1 1 0 1 0

XOR

1 1 1 1 0 0 0 0 ⊕ 1 1 1 1 0 0 0 0 = 0 0 0 0 0 0 0 0

NOT

1 0 1 0 1 0 1 0 = 0 1 0 1 0 1 0 1

¡!

TEOREMA DE DE MORGAN

La principal utilidad de AND y OR es el enmascaramiento XOR se utiliza frecuentemente para hacer cero un resultado, y NOT para la aritmética binaria en complemento a dos

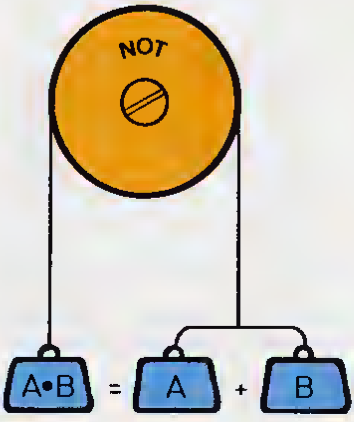
Un importante teorema en el álgebra de Boole es el de *De Morgan*, el cual puede ser formulado de dos formas diferentes: la negación de un AND es igual al OR de las negaciones, o bien, la negación de un OR es igual al AND de las negaciones. Esto se puede expresar mediante la simbología que ya conocemos de la siguiente manera.

$$\overline{A \bullet B} = \overline{A} + \overline{B}$$
$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

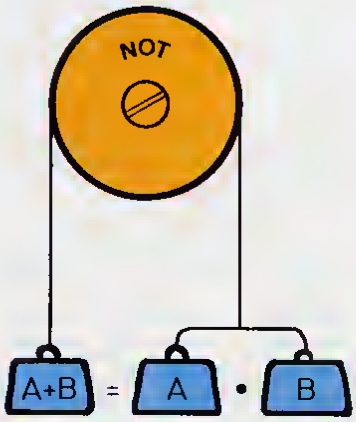
De esta última norma hace uso muy frecuente el *hardware*, puesto que en determinadas circunstancias facilita el diseño de la circuitería del ordenador. Pero todo esto será explicado más profusamente en el siguiente capítulo.

El teorema de De Morgan *puede formularse de dos formas:*
$\overline{A \bullet B} = \overline{A} + \overline{B}$ *ó bien,* $\overline{A + B} = \overline{A} \bullet \overline{B}$

$\overline{A} \bullet \overline{B} = \overline{A} + \overline{B}$



$\overline{A} + \overline{B} = \overline{A} \bullet \overline{B}$



Las operaciones lógicas byte a byte se efectuan como operaciones bit a bit multiples, entre los bits que ocupen posiciones iguales en los operandos

*

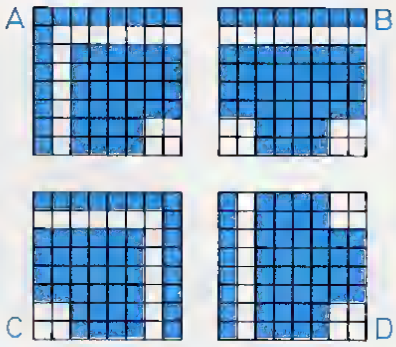
La tecnica que permite asegurar la anulación o activación, de determinados bits del resultado en una operacion lógica, se denomina ENMASCARAMIENTO

*

Las dos posibles formas de enunciar el teorema de *De Morgan* son $\overline{A \bullet B} = \overline{A} + \overline{B}$ y $\overline{A + B} = \overline{A} \bullet \overline{B}$

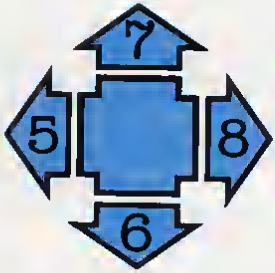
PUZZLE

NUESTRO programa Puzz es la versión informatizada de un clásico pasatiempo. El objetivo final de este juego es ordenar un alfabeto, encerrado dentro de un bastidor de 5 × 5 posiciones.
En el programa, dentro de este cuadrado, cada letra ocupa una posición de carácter, y una de las posiciones está desocupada. El juego nos permite desplazar el espacio en blanco a cualquier posición contigua, de forma que la letra que hubiera allí pase a ocupar la antigua situación del espacio. Gracias a esto, podemos ir desplazando las letras, para conseguir ordenar el alfabeto en el menor número posible de jugadas.
Cualquiera de las letras puede desplazarse a la posición en blanco, siempre que la situación de ésta y la del espacio sean adyacentes.



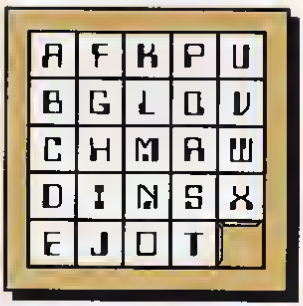
Caracteres gráficos para la definición del bastidor.

COMO JUGAR



Para el movimiento del espacio utilizaremos las teclas 5, 6, 7 y 8

Al ejecutar el programa, aparecerá en la pantalla el alfabeto ordenado. Seguidamente, nuestro Spectrum se encargará de desajustar las posiciones originales de las letras. Este sistema asegura que nosotros podamos resolver el problema, al menos en el número de jugadas que la máquina ha utilizado para desordenar el puzzle, dado que, lógicamente, siempre podríamos seguir los movimientos realizados por ella, aunque en orden inverso.
Para desplazar la posición en blanco, utilizaremos las teclas 5, 6, 7 y 8 correspondientes a izquierda, abajo, arriba y derecha, respectivamente. Este sistema de movimiento implica que no nos podremos desplazar en direcciones diagonales, sino sólo en horizontal y vertical.
Por cada vez que efectuamos una jugada, el programa comprueba mediante una de sus rutinas la resolución del problema. Si hemos conseguido recomponer el puzzle, se detendrá de forma inmediata con un mensaje de enhorabuena. Además, disponemos de un contador interno de movimientos, establecido en la variable CONT, que le permitirá informarnos del número de jugadas en que hemos obtenido la resolución.



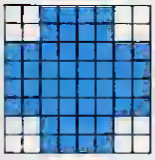
Esta es la forma que debe adoptar el puzzle tras su resolución.

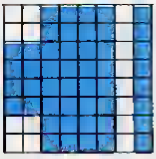
¡!

El programa lo grabaremos usando el siguiente comando **SAVE "PUZZ"**. En el caso de que quisiéramos grabarlo con autoejecucion, teclearíamos **SAVE "PUZZ" LINE 10**

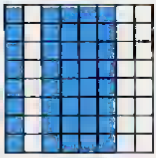
*

La introducción de gráficos implica el paso previo al cursor G (**CAPS SHIFT + 9**) para a continuacion pulsar la tecla cuyo carácter principal es el subrayado

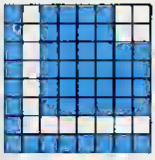
E 

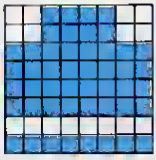

F

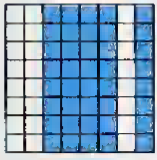
I 

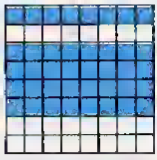

J

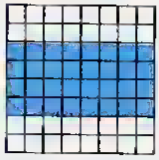
M 

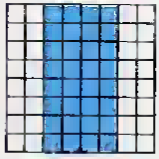
G 


H

K 


L


N


O

EL PROGRAMA

La estructura del programa es lineal. Para la composición del puzzle se recurre a un procedimiento general aplicable a cualquier otro programa de naturaleza semejante. Al intentar construir el planteamiento de un problema resoluble por medio de habilidad o de lógica, hemos de tener en cuenta la condición primaria de que el problema ha de disponer de solución; esto es hemos de construir la estructura inicial a partir de una solución preestablecida.

En este programa, se parte de una matriz ordenada y se descompone por medio de varios accesos a la subrutina de movimiento con valores aleatorios; de esta manera, queda asegurada la existencia de al menos una solución.

Un ejemplo típico de la adopción de este método es la creación de laberintos. Según este sistema, siempre hay que partir de un camino inicial, construyendo a partir del mismo el resto del laberinto, con lo cual queda siempre asegurada la existencia de esa vía de salida.

```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM *   PUZZ      1985    *
 50 REM ************************
 60 BORDER 0
 70 PAPER 0
 80 LET SW=0
 90 INK 9
100 RANDOMIZE
110 LET P=1: CLS
120 LET CONT=0
130 GO SUB 800
140 DIM O$(5,5): DIM M$(5,5)
150 FOR I=1 TO 5
160 FOR J=1 TO 5
170 IF I=5 AND J=5 THEN  LET M$(I,J)=" ": LET O$(I,J
)=" ": LET P=2: GO TO 200
180 LET M$(I,J)=CHR$ (I+5*J+59-32*(I=5 AND J=5))
190 LET O$(I,J)=CHR$ (I+5*J+59-32*(I=5 AND J=5))
200 PRINT  PAPER P;AT 2*I+5,2*J+9;M$(I,J)
210 NEXT J
220 NEXT I
230 LET F=5
240 LET FN=5
250 LET C=5
260 LET CN=5
270 PRINT AT 19,6; FLASH 1; PAPER 4;"   DESORDENANDO
"
280 FOR I=1 TO 250
290 LET X$=STR$ (INT (RND*4)+5)
300 BEEP .01,40
310 GO SUB 460
320 NEXT I
330 PRINT AT 19,6; PAPER 0;"
340 LET SW=1
350 PRINT AT 21,0;"PULSA UNA TECLA PARA COMENZAR"
360 PAUSE 0
370 PRINT AT 21,0;"
380 GO SUB 1030
390 GO SUB 410
400 GO TO 390
410 IF INKEY$<>"" THEN  GO TO 410
420 IF INKEY$="" THEN  GO TO 420
430 LET X$=INKEY$
440 IF CODE X$<53 OR CODE X$>56 THEN  GO TO 410
450 LET CONT=CONT+1
460 IF X$="5" AND C=1 THEN  RETURN
470 IF X$="6" AND F=5 THEN  RETURN
480 IF X$="7" AND F=1 THEN  RETURN
490 IF X$="8" AND C=5 THEN  RETURN
500 BEEP .05,40
510 IF X$="5" THEN  LET CN=C-1
520 IF X$="6" THEN  LET FN=F+1
530 IF X$="7" THEN  LET FN=F-1
540 IF X$="8" THEN  LET CN=C+1
550 LET X$=M$(FN,CN)
560 LET M$(FN,CN)=M$(F,C)
570 LET M$(F,C)=X$
580 PRINT  PAPER 2;AT 2*FN+5,2*CN+9;M$(FN,CN)
590 PRINT  PAPER 1;AT 2*F+5,2*C+9;M$(F,C)
600 LET F=FN
610 LET C=CN
620 IF SW=1 THEN  GO SUB 1090
630 RETURN
640 REM  GRAFICOS USUARIO
650 DATA 255,128,191,191,191,191,188,188
660 DATA 255,0,255,255,255,255,60,60
670 DATA 255,1,253,253,253,253,61,61
680 DATA 188,188,191,191,191,191,188,188
690 DATA 60,60,255,255,255,255,60,60
700 DATA 61,61,253,253,253,253,61,61
710 DATA 188,188,191,191,191,191,128,255
720 DATA 60,60,255,255,255,255,0,255
730 DATA 61,61,253,253,253,253,1,255
740 DATA 188,188,188,188,188,188,188,188
750 DATA 61,61,61,61,61,61,61,61
760 DATA 255,0,255,255,255,255,0,0
770 DATA 0,0,255,255,255,255,0,255
780 DATA 0,0,255,255,255,255,0,0
790 DATA 60,60,60,60,60,60,60,60
800 RESTORE 650
810 LET A$="A8CDEFGHIJKLMNO"
820 FOR F=1 TO LEN A$
830 FOR K=0 TO 7
840 READ A
850 POKE USR A$(F)+K,A
860 NEXT K
870 NEXT F
880 REM  DIBUJO TABLERO
890 PAPER 2: INK 6
900 PRINT AT 6,10;"AL8L8L8L8LC"
910 PRINT AT 7,10;"J O O O O K"
920 PRINT AT 8,10;"DNENENENENF"
930 PRINT AT 9,10;"J O O O O K"
940 PRINT AT 10,10; "ONENENENENF"
950 PRINT AT 11,10; "J O O O O K"
960 PRINT AT 12,10; "ONENENENENF"
970 PRINT AT 13,10; "J O O O O K"
980 PRINT AT 14,10; "DNENENENENF"
990 PRINT AT 15,10; "J O O O O K"
1000 PRINT AT 16,10; "GMHMHMHMHMI"
1010 PAPER 0: INK 9
1020 RETURN
1030 REM  INDICA MOVIMIENTOS
1040 PRINT AT 0,0; PAPER 5;" 5"; INK 9;" IZDUDA. "
1050 PRINT AT 3,0; PAPER 2;" 8"; INK 9;" DERECHA "
1060 PRINT AT 0,20; PAPER 7;" 7 "; INK 9;" ARRIBA "
1070 PRINT AT 3,20; PAPER 6;" 6 "; INK 9;" ABAJO  "
1080 RETURN
1090 REM  COMPROBACION
1100 FOR K=1 TO 5
1110 FOR Q=1 TO 5
1120 IF O$(Q,K)<>M$(Q,K) THEN  RETURN
1130 NEXT Q
1140 NEXT K
1150 PRINT AT 19,8;"LO CONSEGUISTE EN"
1160 PRINT AT 21,10;CONT;"  JUGADAS"
1170 IF INKEY$="" THEN  GO TO 1170
1180 RUN
```